Czech ACM Student Chapter

University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague

Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

CTU Open 2026 is proud to feature Míša, a young and promising Ph.D. student whose everyday life is anything but ordinary. Whether he is walking home from school, decorating the trees in his garden, herding cows on a pasture, hunting vampires in his attic, or just trying to fall asleep, Míša keeps stumbling upon problems that would challenge even the most seasoned contestants. We followed Míša closely and collected a set of the most typical tasks that keep him busy day and night. With open mind, strong determination and some luck, you can help Míša through all of his adventures, and maybe even discover that his life is not so different from yours. We wish your results in the Contest to be even more excellent and outstanding.

Your programs can be written in C, C++, Java, Python or Kotlin programming languages. The choice is yours but you will be fully responsible for the correctness and efficiency of your solutions. All we need is the correct answer produced by your code in some appropriate time.

All programs will read text from the standard input only. The results must be written to the standard output. You are not allowed to use any other files, communicate over network, or create processes. Input and output formats are described in problem statements and must be strictly followed. Values given on one line are separated by one space, if not specified otherwise. Do not print anything more than required. Each printed text line (including the last one) should be terminated by a newline character ("`\n`"), which is not considered to be a part of that line.

Good luck, thank you for coming to our contest, and see you again in 2027!

HUAWEI

icpc diamond
multi-regional sponsor

icpc global sponsor
programming tools

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Accident

`accident.(c|cpp|py)`, `Accident.(java|kt)`

Míša was walking home from school when he tripped on the pavement and dropped his right-angled triangle on the ground. The fall damaged the triangle, and it may no longer be right-angled. Now he wants to fix it with minimal effort — by moving just one of its vertices so that it becomes right-angled again.

Help Míša decide the minimum distance he needs to move one of the vertices, so the triangle becomes right-angled again. It is guaranteed that the input points are not collinear (i.e., they do not lie on a single line) and no two points are the same.

Input Specification

The input consists of three lines, each describing one vertex of the triangle. Each line contains two space-separated integers x and y ($-100 \leq x, y \leq 100$), the coordinates of a point.

Output Specification

Output a decimal number, the minimum distance that one of the points needs to be moved so that this point and the other two points form the vertices of a right-angled triangle.

Your answer will be considered correct if its absolute error does not exceed 10^{-5}.

Sample Input 1

```
4 0
0 4
-1 -1
```

Output for Sample Input 1

```
1.414213562
```

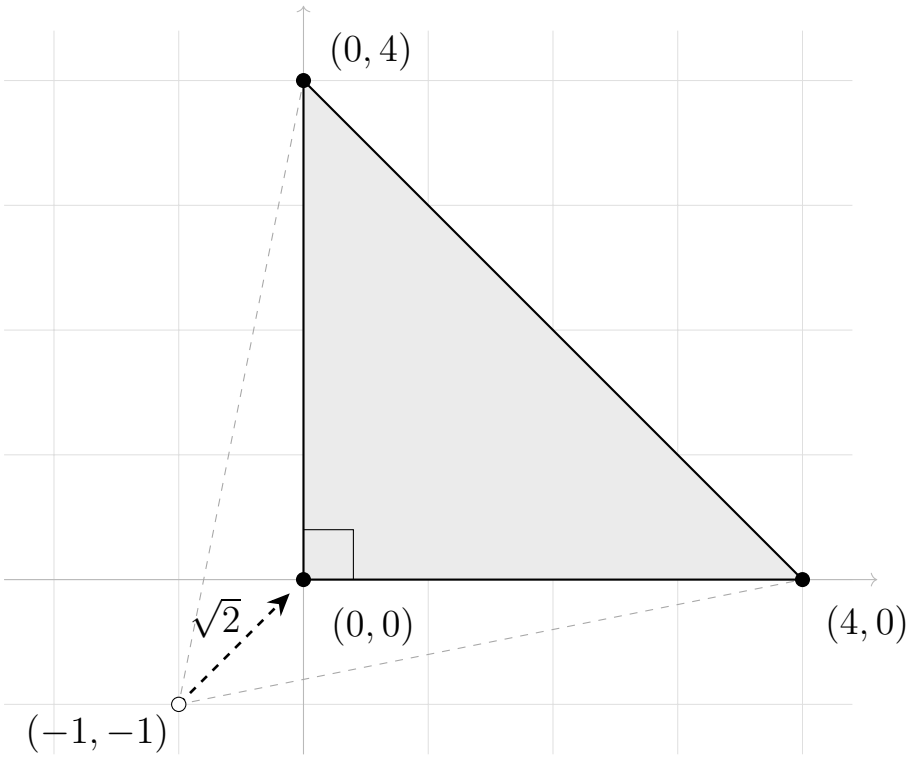
Sample Input 2

```
10 13
14 -4
-42 31
```

Output for Sample Input 2

```
9.013726194
```

In the first sample the cheapest fix is to slide the point $(-1, -1)$ by $\sqrt{2} \approx 1.41421$ towards the origin, where it forms a right angle with $(4, 0)$ and $(0, 4)$.



Sample Input 1

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Birthday

`birthday.(c|cpp|py)`, `Birthday.(java|kt)`

For his birthday, Míša received a small mathematician's kit: a prime number p, and a division machine. The machine can be loaded with a non-negative integer with at most k digits. Míša can then spin the crank, and the machine lights up if the loaded number is divisible by p.

Míša accidentally broke the machine, and now it only lights up when the loaded number is also a palindrome. Still, he gets excited every time the machine lights up for a number that it has never lit up for before.

How many times will Míša be excited? More precisely, for how many distinct integers will the division machine light up?

A palindrome is any integer that reads the exact same forwards and backwards when written in the most succinct format (without zeroes appearing in front of the first nonzero digit).

Input Specification

The input consists of one line with two space-separated integers k and p ($1 \leq k, p \leq 2000$). It is guaranteed that p is a prime.

Output Specification

Output a single integer, the number of distinct integers that the division machine will light up for. As the number can be large, output it modulo $10^9 + 7$.

Sample Input 1	Output for Sample Input 1
`2 3`	`7`

Sample Input 2	Output for Sample Input 2
`67 67`	`943679912`

In the first sample, the palindromes divisible by three with at most 2 digits are $0, 3, 6, 9, 33, 66, 99$.

This page is intentionally left blank.

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Cables

`cables.(c|cpp|py)`, `Cables.(java|kt)`

Míša is building an efficient computer network and wants to test it under load. His network consists of n computers, which are connected using k different types of cables. All cables of the same type must transmit the same amount of data. Each cable can transmit data only in one direction. Míša is able to transmit any amount of data through the cables. He is also able to reverse the direction of all the cables of the same type. He would like to transmit data in such a way that each computer sends out the same amount of data as it receives, and at least one cable transmits a non-zero amount of data.

The types of cables have a fixed defined order. Míša selects a contiguous subsequence of those types. Then for each selected cable type he selects the direction and amount of data going through. After the selection the data flow has to be balanced at each computer. Help Míša find the shortest such subsequence.

More formally, Míša is given a sequence of directed edge sets $E_1, E_2, \ldots, E_k$, each E_i representing the cables of type i. A contiguous nonempty block $E_i, E_{i+1}, \ldots, E_j$ is *balanced* if Míša can, for each type $q \in \{i, i+1, \ldots, j\}$, choose whether to reverse all edges of E_q and choose a non-negative real amount x_q, such that

- at least one of the amounts $x_i, x_{i+1}, \ldots, x_j$ is non-zero, and
- when every edge of type q transmits the amount x_q in its chosen direction, the total incoming amount equals the total outgoing amount at each computer.

Equivalently, the amounts x_q may be arbitrary real numbers (not all zero), where a negative amount means that the edges of type q are reversed.

Your task is to find a balanced block of minimum length $j-i+1$, or determine that no balanced block exists.

Input Specification

The first line contains two integers n and k ($2 \le n \le 100$, $1 \le k \le 2 \cdot 10^5$), the number of computers and the number of cable types.

The k cable types follow, in order; the i-th of them is the edge set E_i. Each type is described by a line containing the number of cables $|E_i|$ ($|E_i| \ge 1$), followed by $|E_i|$ lines, each containing two integers u and v ($1 \le u, v \le n$, $u \ne v$), meaning that a cable of this type transmits data from computer u to computer v. The computers are numbered 1 to n.

Within a single type no two cables share the same ordered pair (u, v), and no type contains both cables (u, v) and (v, u) for any pair u, v.

The total number of cables satisfies $|E_1| + |E_2| + \cdots + |E_k| \le 10^6$.

Output Specification

Output a single integer: the smallest number of consecutive cable types Míša can select so that the data can be transmitted in the balanced way, that is, the minimum length $j - i + 1$ of a balanced block $E_i, E_{i+1}, \ldots, E_j$.

If no balanced block exists, output -1.

Sample Input 1

```
3 3
2
1 2
2 3
2
2 3
3 1
2
1 3
2 1
```

Output for Sample Input 1

```
3
```

Sample Input 2

```
3 3
1
1 2
1
2 3
2
3 1
3 2
```

Output for Sample Input 2

```
3
```

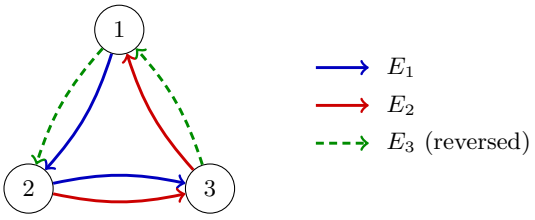
Sample Input 3

```
4 3
2
1 2
2 3
1
3 4
2
4 1
1 2
```

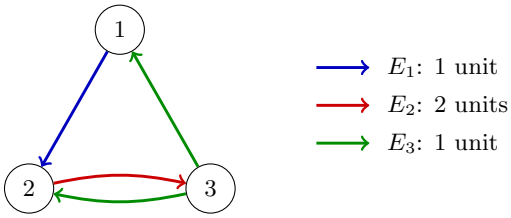
Output for Sample Input 3

```
-1
```

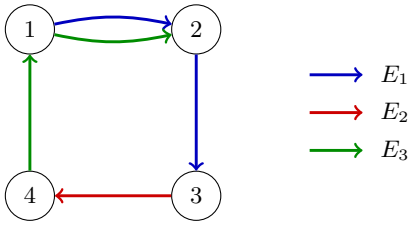
In the first sample the shortest balanced block is all three types, each used once with E_3 reversed.



In the second sample the shortest balanced block is all three types, none reversed, with E_2 carrying two units of data and E_1, E_3 one unit each.



In the third sample no block is balanced, even though together the cables go around the cycle $1 \to 2 \to 3 \to 4 \to 1$. At computer 2 the two cables of E_1 cancel out, so the cable $1 \to 2$ of E_3 cannot be balanced and E_3 must carry nothing. Then computer 4 is touched only by E_2, and after that computer 1 only by E_1, so all amounts are zero.



This page is intentionally left blank.

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Housing

`housing.(c|cpp|py)`, `Housing.(java|kt)`

Míša with his above-average Ph.D. salary is planning to buy a house. The house is to be located on a development street where houses are numbered from 1 to n. To curate his perfect neighborhood, he plans to demolish a contiguous (possibly empty) block of houses from the far left of the street, and another (possibly empty) block from the far right.

This leaves a single, contiguous segment of remaining houses. Míša requires the price of the house to sit in the middle of the market — meaning its price must be the exact median of this newly formed neighborhood. More precisely, he can live in the i-th house ($i \in \{1, \ldots, n\}$) if there exists a contiguous row of houses $\ell, \ell+1, \ldots, r-1, r$ such that house i lies in this row (i.e., $\ell \leq i \leq r$), the length $r-\ell+1$ of the row is odd and at least 3, and the price a_i of the i-th house is the median of the prices $a_\ell, a_{\ell+1}, \ldots, a_{r-1}, a_r$.

Recall that an element x of an array $b = (b_1, b_2, \ldots, b_{t-1}, b_t)$ of odd length t is the *median* if, after sorting the array b, the value x lies at position $\frac{t+1}{2}$.

It is guaranteed that every house on the street has a unique price.

Help Míša figure out which houses are viable options for living given his requirements.

Formally, determine which elements a_i of an array $a_1, a_2, \ldots, a_n$ are the median of some subarray of odd length at least 3 containing a_i. All the elements of the array are distinct.

Input Specification

The first input line contains an integer n ($3 \leq n \leq 2 \cdot 10^5$), the number of houses in the street. The next input line contains n pairwise distinct integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$), i.e., a permutation of $1, \ldots, n$, representing the prices of the houses on the street.

Output Specification

Output a single line with n space-separated strings. The i-th string is either `YES` if Míša can live in the i-th house (in the house with price a_i), otherwise, the string is `NO`.

Sample Input 1

```
5
2 5 1 4 3
```

Output for Sample Input 1

```
YES NO NO YES YES
```

This page is intentionally left blank.

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Lanterns

`lanterns.(c|cpp|py)`, `Lanterns.(java|kt)`

Míša is a passionate lover of gardening and also loves graph theory. That is why he grows trees in his modest garden. Recall that a *tree* is a connected undirected acyclic graph. He likes to decorate his trees with lanterns. However, each vertex of the tree must either be directly illuminated (have a lantern on it) or have a neighboring vertex with a lantern. Formally, the set of lanterns must form a *dominating set*. A vertex set $S \subseteq V$ of a tree $T = (V, E)$ is a *dominating set* if every vertex of T is either in S, or has a neighbor in S. He has a fixed budget k for all of his trees — every tree is illuminated with exactly k lanterns.

Lately, he noticed that his illuminating arrangements are not so uniform; depending on the night, he might want the garden to look completely different. He would like to transform one configuration of lanterns, represented by a dominating set $S_1 \subseteq V$, into a (possibly different) dominating set $S_2 \subseteq V$, using the following operations.

In one operation, Míša can take any positive number of lanterns from any vertex v (at most as many as there are on v) and move them to a neighbor of v. Note that it is possible for a vertex to be occupied by more than one lantern after some operations. In the beginning, exactly one lantern is placed on every vertex of S_1, and the goal is to have exactly one lantern placed on every vertex of S_2. After each operation, it must be ensured that the tree never falls into darkness: the set of vertices occupied by at least one lantern must always form a dominating set of T.

Help Míša determine the minimum number of lantern movements needed to achieve his goal, or tell him that it is impossible.

Input Specification

The first input line contains one integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of vertices in the tree T. The vertices are labeled $0, 1, \ldots, n-1$. Next, $n-1$ lines follow, each of them contains two integers u, v ($0 \leq u, v \leq n-1, u \neq v$), labels of endpoints of an edge of T. The next line contains an integer k ($1 \leq k \leq n$), the number of lanterns on Míša's tree. Next, there is a line describing the dominating set S_1, followed by a line describing dominating set S_2. Each of these lines contains k pairwise distinct labels of vertices in the respective dominating set. It is guaranteed that both S_1 and S_2 are dominating sets of T.

Output Specification

If no solution exists, i.e., there is no sequence of admissible operations transforming S_1 to S_2, output a single line with the number -1. Otherwise, output the following.

The first line of output contains an integer c, the number of commands. Next, there are c lines, each contains one command. A command is a triple u, v, ℓ, separated by spaces, where u, v are adjacent vertices of T and ℓ is a positive integer. The command represents moving

ℓ lanterns from u to v. Note that prior to the command execution, there must be at least ℓ lanterns available on u and after the movement, the set of vertices occupied by lanterns must form a dominating set of T. The total number of lantern moves is the sum of values ℓ over all c commands. This sum must be minimum possible. Note that there might exist different optimal solutions with different number of commands, but the sum of ℓ over the commands is the same. If there are multiple optimal solutions, you may output any of them.

It can be shown that if a solution exists, then there also exists a solution that uses at most $10 \cdot n$ commands. Your solution will be rejected if you output more than $10 \cdot n$ commands.

Sample Input 1

```
3
0 1
1 2
2
0 1
1 2
```

Output for Sample Input 1

```
2
0 1 1
1 2 1
```

Sample Input 2

```
7
0 2
1 2
2 3
3 4
4 5
4 6
4
0 1 2 4
2 4 5 6
```

Output for Sample Input 2

```
6
1 2 1
0 2 1
2 3 2
3 4 2
4 6 1
4 5 1
```

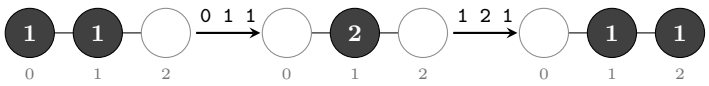


Figure 1: An optimal solution for Sample Input 1. The total number of lantern movements is $1 + 1 = 2$.

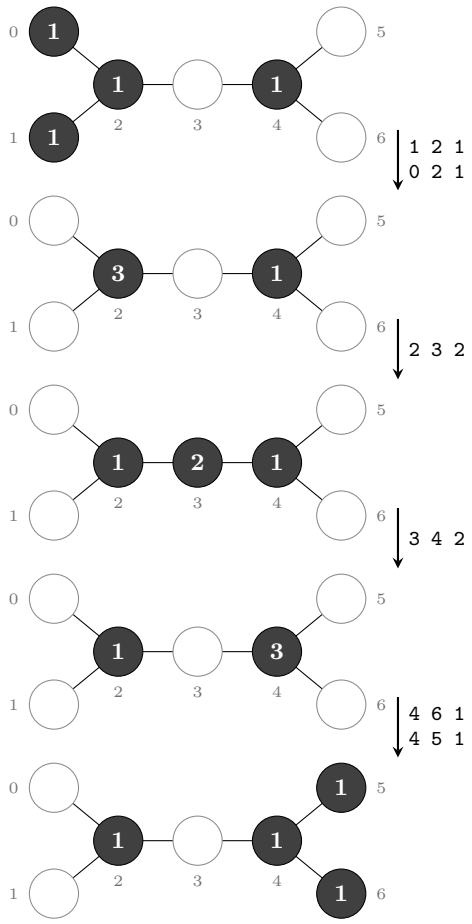


Figure 2: An optimal solution for Sample Input 2. The total number of lantern movements is $1 + 1 + 2 + 2 + 1 + 1 = 8$.

This page is intentionally left blank.

Czech ACM Student Chapter

University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague

Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Milky way

`milkyway.(c|cpp|py)`, `Milkyway.(java|kt)`

Míša hasn't lived in the real world for a long time. Instead, he sits in a dark room with heavy studio headphones clamped tightly to his ears, listening to nothing but static clicks and high-pitched cosmic squeals. Míša is capturing signals from deep space. Every second, exactly one tone arrives at a specific frequency.

The universe is a malicious entity, and Míša suffers from a highly specific form of intergalactic OCD. He cannot tolerate chaos. The moment he puts his headphones on, his brain demands absolute, predictable harmony. Hence, Míša is looking for satisfactory sequences of tones.

A contiguous sequence of tones is *satisfactory* for Míša if the frequencies of the tones are non-decreasing. More formally, the sequence $a_\ell, a_{\ell+1}, \ldots, a_{r-1}, a_r$ is satisfactory if $a_\ell \leq a_{\ell+1} \leq \cdots \leq a_{r-1} \leq a_r$.

Míša would like to answer q queries. Each query is given by a pair of endpoints ℓ, r, and he wants to determine whether the sequence $a_\ell, a_{\ell+1}, \ldots, a_{r-1}, a_r$ is satisfactory or not.

Input Specification

The input starts with two lines. The first line contains two integers n ($1 \leq n \leq 10^6$) and q ($1 \leq q \leq 10^5$) separated by spaces, giving the number of frequencies and number of queries respectively.

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($-10^6 \leq a_i \leq 10^6$) representing the frequencies Míša has captured.

Then q lines follow, each with two space-separated integers ℓ and r ($1 \leq \ell \leq r \leq n$) indicating the start and end of the query interval $a_\ell, a_{\ell+1}, \ldots, a_{r-1}, a_r$.

Output Specification

For each query, output `YES` if the query interval is satisfactory for Míša, and `NO` otherwise.

Sample Input 1

```
7 5
1 2 3 2 4 -1 -1
1 2
3 4
1 7
1 1
6 7
```

Output for Sample Input 1

```
YES
NO
NO
YES
YES
```

This page is intentionally left blank.

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Nightmare

`nightmare.(c|cpp|py)`, `Nightmare.(java|kt)`

The night was dark and heavy, but for Míša, the true horror was staring into the mirror of his own mind. Sleep refused to come. Instead, shadows began to materialize in the darkness of his room. These shadows, however, did not take human form. Rather, they branched like cold, merciless edges of an ancient graph.

Each edge in this terrifying network pulsed with one of two lights — a bloody red one or a cold, icy blue one. Míša felt the labyrinth swallowing him whole. He knew that if there existed even a single cycle in this graph in which exactly one edge had a different color than all the others, the entire structure would collapse into chaos, and he would never find peace. Trapped in the prison of his own imagination, he had to discover at all costs whether every single cycle of the network was either perfectly monochromatic or contained at least two red and at least two blue edges.

By a *cycle* we mean a simple cycle, i.e., a closed path that does not visit any vertex more than once.

Input Specification

The first input line contains two integers n, m ($1 \leq n, m \leq 3 \cdot 10^5$), the number of vertices and edges of the graph Míša is witnessing. The graph is undirected. The vertices are labeled $0, 1, \ldots, n-1$. Each of the next m lines describes one edge of the graph. An edge is described by a triple of integers u, v, c, separated by spaces. u and v are the labels of the endpoints of the edge, c denotes the color of the edge. Red is denoted by $c = 0$, blue is denoted by $c = 1$. It is guaranteed that the graph is simple, i.e., it contains no multiedges or self-loops.

Output Specification

Output a single line with the text `YES` if Míša will be able to find peace, in other words, if every cycle is either monochromatic (all edges are red or all edges are blue), or it contains at least 2 edges of each color. If Míša will not be able to find peace, output `NO`.

Sample Input 1

```
3 3
0 1 1
0 2 1
1 2 0
```

Output for Sample Input 1

```
NO
```

Sample Input 2

```
4 4
0 1 1
1 2 1
2 3 0
3 0 0
```

Output for Sample Input 2

```
YES
```

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Park

`park.(c|cpp|py)`, `Park.(java|kt)`

Míša was walking through a park and saw a beautiful tree – an undirected acyclic connected simple graph. He noticed many squirrels running around the tree. He was observing their behavior and noticed that they like to run along some paths more than others. Using his mathematical skills, he found out the formula that decides how much the squirrels like each path.

First, he noticed that the tree has N vertices and each of them has a unique beauty value from 1 to N. He then defined the popularity of a path as its length (number of edges in the path) multiplied by the maximum beauty value over all vertices in the path.

Formally, let b_v denote the beauty value of vertex v. The popularity of a path visiting vertices $p_1, p_2, \ldots, p_k$ is

$$(k-1) \cdot \max(b_{p_1}, b_{p_2}, \ldots, b_{p_k}).$$

Míša wants to know the total popularity of all paths in the tree, that is, the sum of popularities of the paths between all unordered pairs of distinct vertices (a path and its reverse are counted only once). Help him compute it quickly. Since the value can be large, output it modulo $10^9 + 7$.

Input Specification

The first line contains one integer N ($1 \leq N \leq 10^5$), the number of vertices.

The second line contains N space-separated integers $b_1, b_2, \ldots, b_N$, the beauty values of vertices $1, 2, \ldots, N$. In other words, b_i is the beauty value of vertex i. The values $b_1, b_2, \ldots, b_N$ form a permutation of $1, 2, \ldots, N$.

Each of the following $N-1$ lines contains two space-separated integers u and v ($1 \leq u, v \leq N$, $u \neq v$), describing an edge between vertices u and v. The edges are guaranteed to form a tree.

Output Specification

Output a single integer: the total popularity of the tree, that is, the sum of the popularities of the paths over all unordered pairs of distinct vertices, taken modulo $10^9 + 7$.

Sample Input 1

```
3
1 2 3
1 2
2 3
```

Output for Sample Input 1

```
11
```

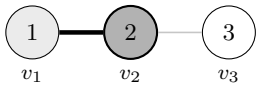
Sample Input 2

```
6
4 1 6 2 5 3
1 2
2 3
3 4
3 5
5 6
```

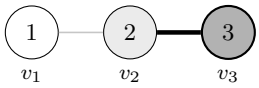
Output for Sample Input 2

```
183
```

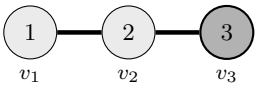
In the first sample, the tree is made of a path $v_1 - v_2 - v_3$ with beauty values $b = (1, 2, 3)$. It contains three paths which contribute their popularity as shown below; bold edges trace each path and the shaded vertex carries the largest beauty on it.



$\{1, 2\}: \ 1 \cdot \max(1, 2) = 2$



$\{2, 3\}: \ 1 \cdot \max(2, 3) = 3$



$\{1, 3\}: \ 2 \cdot \max(1, 2, 3) = 6$

Total popularity $= 2 + 3 + 6 = 11$

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Rope

`rope.(c|cpp|py)`, `Rope.(java|kt)`

Míša was taking a peaceful afternoon stroll through the countryside when he stumbled upon a curious sight: a pasture dotted with stationary cows. By total coincidence — and for reasons he refuses to explain — Míša happened to be carrying a sturdy piece of rope of fixed length r. Never one to pass up a strange challenge, Míša wondered: if he tied the ends of the rope together to form a loop, how many cows could he herd together inside it? Help Míša figure out the maximum number of cows he can capture within a single loop of his rope.

Formally, the cows are points in the plane. Míša lays the rope on the ground as a closed curve of length at most r. The rope does not have to be stretched into a nice loop: it may touch or overlap itself, for example run along a segment and back. A cow is captured if it lies inside the area enclosed by the rope or directly on the rope. Find the maximum number of cows that can be captured.

In particular, a single cow can always be captured (the rope can be coiled around a single point), and two cows at distance d can be captured with a rope of length $2d$ by laying the rope along the segment between them and back (see Sample Input 2).

Input Specification

The first line contains two space-separated integers n and r ($3 \le n \le 100$, $1 \le r \le 10000$), the number of cows and the maximum length of the rope. Each of the next n lines contains two space-separated integer coordinates x and y of a cow ($-100 \le x, y \le 100$). It is guaranteed that no two points are equal and no three lie on a common line.

Guarantee: Replacing r with any value in $[r - 10^{-5}, r + 10^{-5}]$ does not change the maximum number of cows that can be captured.

Output Specification

Output a single line with an integer, the maximum number of cows Míša can capture.

Sample Input 1

```
4 7
0 2
2 0
2 2
0 0
```

Output for Sample Input 1

```
3
```

Sample Input 2

```
3 3
1 1
2 0
0 0
```

Output for Sample Input 2

```
2
```

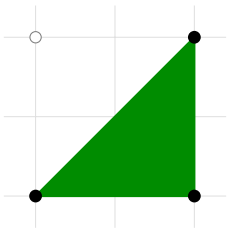


Illustration for sample input 1. The curve length is $4 + 2\sqrt{2} \leq r = 7$

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Shopping

`shopping.(c|cpp|py)`, `Shopping.(java|kt)`

Míša loves numbers, so he bought n of them from a convenience store. As a true number enthusiast, he wants to find the bitwise OR of all possible subset sums of these numbers.

Formally, let $A = \{a_1, a_2, \ldots, a_n\}$ be the multiset of the numbers Míša bought. Míša wants to compute the expression

$$\bigoplus_{S \subseteq A} \left(\sum_{a \in S} a \right)$$

where $\bigoplus$ represents the bitwise OR. Help Míša with this task. Note that the sum for the empty set is 0, i.e., $\sum_{a \in \emptyset} a = 0$.

Input Specification

The first input line contains an integer n ($1 \leq n \leq 10^5$), the number of elements in Míša's multiset A. The second line contains n space-separated integers a_i ($1 \leq a_i \leq 10^{18}$). It is guaranteed that the sum of all elements in A does not exceed 10^{18}.

Output Specification

Output a single integer, the bitwise OR over all possible subset sums of the numbers.

Sample Input 1

```
2
3 5
```

Output for Sample Input 1

```
15
```

Sample Input 2

```
3
4 9 13
```

Output for Sample Input 2

```
31
```

Sample Input 3

```
3
2 9 32
```

Output for Sample Input 3

```
43
```

In the first sample, the subsets are $\{\}, \{3\}, \{5\}, \{3, 5\}$, giving sums $0, 3, 5, 8$. In binary, these are $0000_2, 0011_2, 0101_2, 1000_2$, and their bitwise OR is $0000_2 \mid 0011_2 \mid 0101_2 \mid 1000_2 = 1111_2 = 15$.

In the second sample, the subset sums are $0, 4, 9, 13, 17, 22, 26$, and their bitwise OR is 31.

In the third sample, the subset sums are $0, 2, 9, 11, 32, 34, 41, 43$, and their bitwise OR is 43.

This page is intentionally left blank.

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Trains

`trains.(c|cpp|py)`, `Trains.(java|kt)`

Míša is looking down at a railway station with m parallel straight tracks, each holding exactly one train. Each track is an infinite line of unit-length cells with its own integer coordinates. Míša can only see the cells numbered $0, 1, \ldots, n-1$ of each track.

Each train is described by four values: the position p_i of its leftmost cell at time 0, its speed s_i, its direction $d_i \in \{\texttt{L}, \texttt{R}\}$, and its length in cells ℓ_i.

The train on track i occupies ℓ_i consecutive cells, which at time 0 are $p_i, p_i+1, \ldots, p_i+\ell_i-1$. It moves as a single rigid block at constant speed s_i cells per unit of time, in direction $d_i \in \{\texttt{L}, \texttt{R}\}$ (`R` means increasing coordinates, `L` means decreasing coordinates). Thus, at time t the train occupies cells $p_i + v_i \cdot t, \ldots, p_i + v_i \cdot t + \ell_i - 1$, where $v_i = s_i$ if $d_i = \texttt{R}$ and $v_i = -s_i$ if $d_i = \texttt{L}$.

The engine always occupies a cell at the front of the train: cell $p_i + v_i \cdot t + \ell_i - 1$ if $d_i = \texttt{R}$, or cell $p_i + v_i \cdot t$ if $d_i = \texttt{L}$. Every other cell of the train is a car.

Given the time t, determine what Míša sees: for each track and each visible cell $0, \ldots, n-1$, indicate whether the cell is empty, holds the train engine or a train car.

Input Specification

The first input line contains three integers n, m, t ($1 \le n, m \le 10^5$, $n \cdot m \le 4 \cdot 10^6$, $0 \le t \le 10^4$), the number of visible cells in each track, the number of tracks, and the current time.

Each of the next m lines describes a train on one track, using four values p_i, s_i, d_i, ℓ_i: the position of the train's leftmost cell at time 0 ($-10^5 \le p_i \le 10^5$), the train speed ($0 \le s_i \le 10^5$), the train direction ($d_i \in \{\texttt{L}, \texttt{R}\}$), and the train length in cells ($1 \le \ell_i \le 10^5$).

Output Specification

Output m lines, one per track in the same order as the input. Line i must contain exactly n printable characters, where the j-th character (for $j = 0, \ldots, n-1$) describes cell j of track i at time t: `!` if the cell holds the engine, `#` if the cell holds a car, and `=` if the cell is empty.

Sample Input 1

```
10 3 3
0 1 R 1
9 1 L 1
7 0 R 2
```

Output for Sample Input 1

```
===!======
======!===
=======#!=
```

Sample Input 2

```
6 2 1
-2 1 R 5
4 0 L 5
```

Output for Sample Input 2

```
###!==
====!#
```

Sample Input 3

```
5 3 3
10 0 R 3
0 1 R 1
-5 2 R 1
```

Output for Sample Input 3

```
=====
===!=
=!===
```

Czech ACM Student Chapter
University of West Bohemia in Pilsen
Masaryk University
Pavol Jozef Šafárik University in Košice

Czech Technical University in Prague
Technical University of Ostrava
University of Žilina
Comenius University

CTU Open Contest 2026

Vampires

`vampires.(c|cpp|py)`, `Vampires.(java|kt)`

Míša decided to stop being afraid of vampires. In his attic, he found triples of bats hanging upside down: two small ones and a large one in every triple. Each bat has a positive integer written on it. Míša wants to check whether the large bat is actually a vampire! The large bat is a vampire if and only if the integers on the three bats satisfy the conditions of *vampirism*.

Help Míša identify the vampires!

A natural number n (the number on the large bat) with $2k$ digits, $n = n_{2k}n_{2k-1}\ldots n_2n_1$, is a *vampire number* with fangs a and b (the numbers on the small bats) if and only if:

- $n = a \cdot b$,
- a and b both have k digits: $a = a_k a_{k-1}\ldots a_2a_1$ and $b = b_k b_{k-1}\ldots b_2b_1$,
- at least one of a or b is not divisible by 10,
- the concatenation of the digits of a and b, $a_k a_{k-1}\ldots a_2a_1b_kb_{k-1}\ldots b_2b_1$, is a permutation of the digits of n.

Input Specification

The first line of the input contains a single integer t ($1 \le t \le 1000$), the number of test cases. Each of the next t lines contains three integers n, a, b ($1 \le n, a, b < 10^8$) describing a triple Míša found.

Output Specification

For each test case, output `YES` if n is a vampire number with fangs a and b, and `NO` otherwise.

Sample Input 1

```
4
1260 21 60
27 81 2187
1210 10 12
231495 345 671
```

Output for Sample Input 1

```
YES
NO
NO
NO
```

This page is intentionally left blank.